Programming Basics bei Max Rheiner

Break Out
Wald zeichnen
Mal-Programm

kleinere Übungen:

Zufallszahlen / Zeichnen
Animation (Linie/Ball)
Rotierende Smileys
Rekursive Funktionen / Bäume

Break Out

Die Endaufgabe bestand darin, vom Spiel *Break Out* ausgehend ein eigenes Spiel zu entwickeln. Das herkömmliche *Break Out* ist sehr abstrakt und bietet unzählige Interpretationsmöglichkeiten. Ich stellte mir die Frage, da *to break out* soviel wie *ausbrechen* bedeutet, woraus man in meinem Spiel ausbrechen kann und in welchen Kontext, bzw. welche Geschichte man das Spiel setzen kann. Wie man im Startfenster sieht, handelt es sich bei mir um eine Höhlen-Situation, in der man wie beschrieben, ausbrechen muss; dazu sind die Steine zu zerstören, die man im Dunkeln findet, sobald ‚p' gedrückt wird. Der Schläger am unteren Fensterrand ist eine Lichtquelle (wahrscheinlich eine Taschenlampe, siehe Game-Over-Mode), genau so wie der Ball, mit dem man versucht die Steine zu zerstören. Hier behält meine Braek-Out-Version sein abstraktes Element. Es wird nicht aufgedeckt, was der Ball ist und wieso er Steinsbrocken zerstören kann.
Neben einem neuen, vierten Game-Mode, dem Win-Fenster, habe ich ausserdem Sound-Samples zu Max's Ausgangscode hinzugefügt. Damit wird die Tropfsteinhöhlenatmosphäre, bzw die Story unterstützt.

Probleme / Fragestellungen bei der Umsetzung:

> Anfangs hatte ich den dunklen Raum nur durch den leuchtenden Ball erhellt. Dies war jedoch nicht ganz ideal gelöst, da man nie, bzw. erst, wenn es zu spät war, wusste, wo sich der Schläger befindet. Deshalb wurde der Schläger zu einer Lichtquelle umgeschrieben.

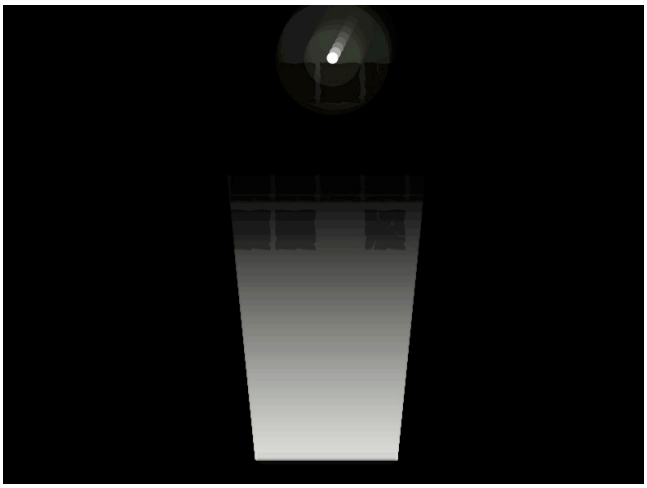
> Weitere Änderungen habe ich bei der Steuerung des Schlägers vorgenommen. Ich bin erst wieder zur Maussteuerung zurückgekehrt, nachdem ich es mit Links-Rechts-Tasten versucht habe. Dies war jedoch zu langsam und stockend, bzw. zu wenig intuitiv. Das Steuern mit der Maus funktioniert besser.

> Am Anfang waren die Bricks nach einem Treffer kabutt. Um das Spiel zu erschweren, habe ich ihre Lebensdauer um zwei weitere Treffer erhöht. Der Zerstörungsgrad ist ihnen anzusehen, da ich je nach „Lebens-Wert" verschiedene Bilder reinlade.

> Eine weitere kleine Spielerei, die ich schliesslich wegliess, war das Ändern der Ballgrösse durch Tastendrücken. Dies hat nicht in den Spielfluss gepasst. Man ist zu beschäftig mit dem Steuern des Schlägers als ‚dass man noch irgendwelche Tasten drücken könnte.

> Download-Link Code Text
http://blogs.iad.zhdk.ch/codingspace/2013/09/25/mi-25-09-2014-start-endaufgabe/

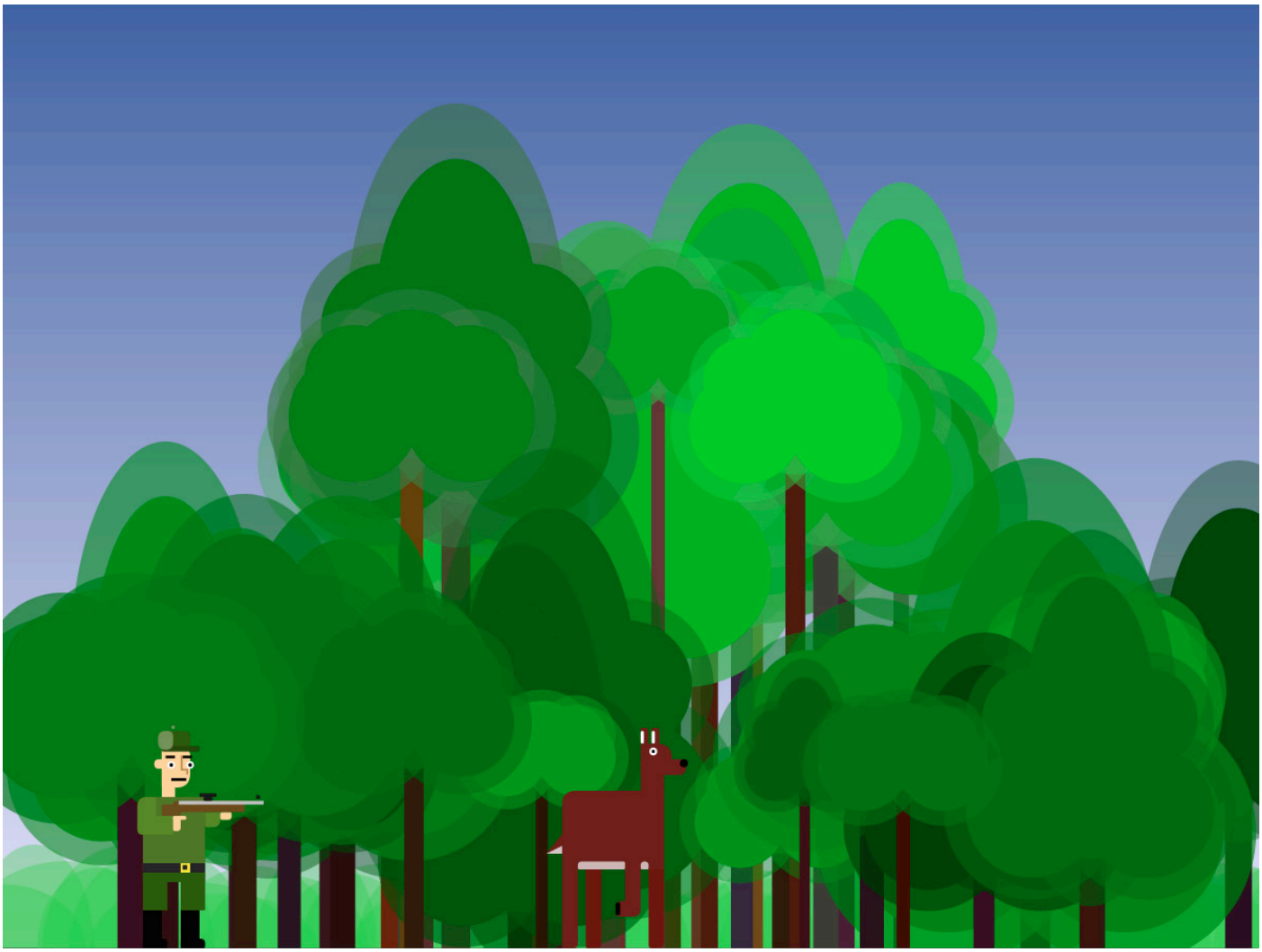
> Start-Menu



> Game-Mode

> Game-Over-Mode
(falls man den Ball nicht abfängt)

> Win-Mode
(sobald alle Steine zerstört wurden)



Wald zeichnen

Mit den Wissen zu Schleifen und dem Verschieben, Rotieren und Skalieren von Koordinaten-systemen, habe ich einen Wald programmiert. Er besteht aus mehreren übereinanderliegenden Ebenen. Das Übereinanderliegen der gezeichneten Ebenen wird im Codetext dadurch erzielt, dass man die Objekte nacheinander auflistet.

1. Himmel mit Farbverlauf

2. Unterholz

3. erste Baumgruppe (in der Mitte des Bildes)

4. zweite Baumgruppe (über die x-Achse verteilt)

5. Reh und Fritz der Jäger

Ich wurde hier gefragt, ob ich das Reh und den Jäger als Bild hineingeladen habe. Dem ist nicht so. Sie sind beide aus abgeschrägten Rechtecken und Ellipsen zusammengesetzt. Dies mag aufwändig erscheinen, doch durch Copy+Paste ist das ziemlich rasch gegangen und ich habe damit eine coole Comic-Ästhetik hingekriegt.

Bei jedem Klick wiederholt sich der Draw-Vorgang. Das heisst die oberen Punkte 2 – 5 werden nochmals gezeichnet. Die Blätterfarben, Baumstammpositionen, -längen und -farben, Positionen von Fritz und dem Reh werden in bestimmten Schranken zufällig verändert. Hier gibt es noch eine kleine Finesse: Fritz erscheint nur, wenn seine x-Koordinate um einen gewissen Wert kleiner ist, als die vom Reh. So ist er immer hinter dem Reh, bzw. links von ihm, um es überhaupt jagen zu können.

Ausserdem haben Fritz und das Wild beide eine Skalier-Variabel, mit der man sie unabhängig voneinander vergrössern oder verkleinern kann.

> Download-Link Code Text
http://blogs.iad.zhdk.ch/codingspace/2013/09/23/fr-20-09-2013/

Mal-Programm

Die Aufgabe war es, ein Mal-Tool zu programmieren. Voraussetzung dafür waren Funktionen, Framework-Events (Mauseingabe, Tasteneingabe, Programmstart...), Bedingungen und andere vorausgehende Themen.

Es ist möglich, mit den Tasten 1 bis 5 die Farbe zu wählen. Sobald eine Farbe gewählt wurde, wird diese auch angezeigt.

Meine ursprüngliche Idee war, ein Mal-Programm mit Knöpfen, die man anklicken kann, um die Farbe zu wählen. Doch ich scheiterte an dem Vorhaben und kehrte dann zurück zur Version, in der man mit den Computertasten die Farben wählt. Mein ersteres Vorhaben wäre anscheinend mit sogenannten ***Guis*** realisierbar gewesen, was ich im Nachhinein erfahren habe.

> Download-Link Code Text
http://blogs.iad.zhdk.ch/codingspace/2013/09/19/do-19-09-2013/

Beim Arbeiten ist folgende Frage aufgetaucht – Wie kann ich die Transparenz beeinflussen bei:

color c1 = (...); ?

Antwort:

color c1 = color (255,200,150,**80**) - der letzte Parameter, also 80, bestimmt die Transparenz

Auch “stockt” der Strich nicht mehr, da ich anstelle von:

```
01 void mousePressed()
02 {
03 fill(c_current);
04 ellipse(mouseX, mouseY, strichdicke, strichdicke);
05 }
06
07 void mouseDragged()
08 {
09 fill(c_current);
10 ellipse(mouseX, mouseY, strichdicke, strichdicke);
11 }
```

Folgendes geschrieben habe:

```
1 void mouseDragged()
2 {
3 stroke(c_current);
4 line(pmouseX, pmouseY, mouseX, mouseY);
5 }
```

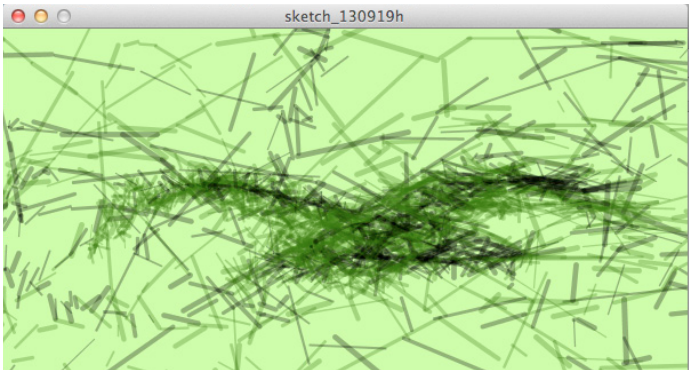
> Download-Link Code Text

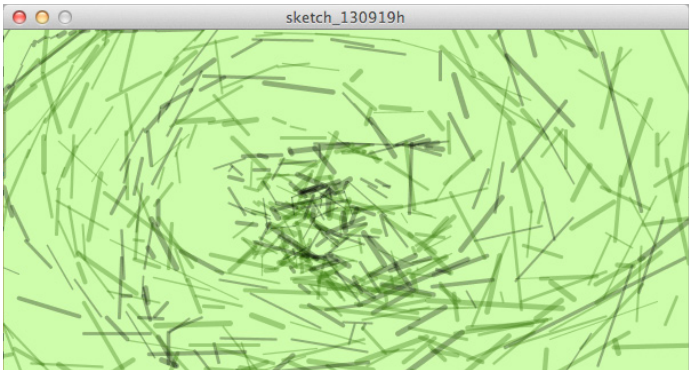
Kleinere Übungen

1 Zufallszahlen / Zeichnen

Bei Mausklick werden Striche mit zufälliger Breite und Ausdehnung gezeichnet. Sobald die Maus ein weiteres Mal gedrückt wird, wechselt die Farbe in einen anderen Grünwert. Dies wiederholt sich bei jedem Klick.

> Download-Link Code Text (ganz unten)
http://blogs.iad.zhdk.ch/codingspace/2013/09/19/do-19-09-2013/

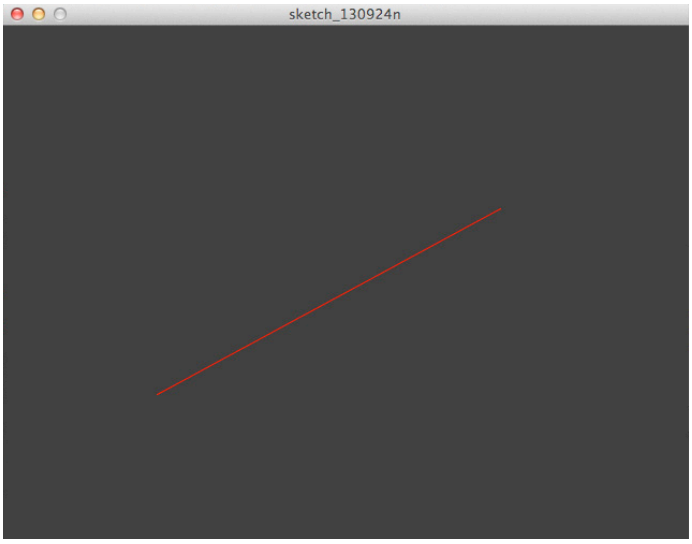


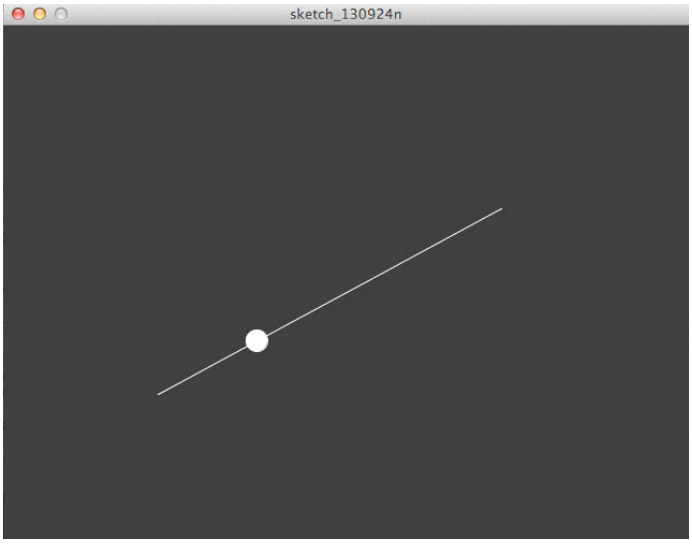


2 Animation

Der Pfad der Animation wird per Mausklick definiert. Nach dem zweiten Klick, bzw. dem Setzten des Pfadendpunktes, bewegt sich der Kreis vor- und rückwerts. Sobald die Maus ein drittes Mal gedrückt wird, setzt man den Anfangspunkt für einen neuen Pfad.

> Download-Link Code Text
blogs.iad.zhdk.ch/codingspace/2013/09/25/mi-25-09-2013-nacharbeiten-arrays-listen-svg-

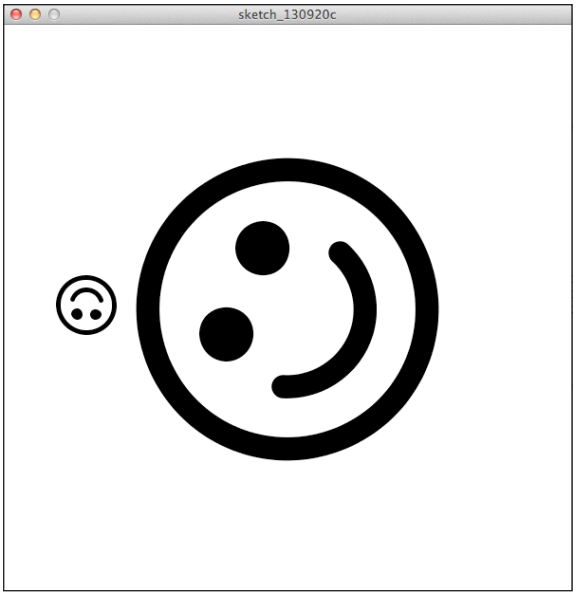




3 Rotierende Smiley

Der kleine Smiley rotiert stehts um den grossen. Dieser wird durch das Verändern der Cursor-Koordinaten gedreht und skaliert. In diesen Funktionen unterliegt der kleine dem grossen Smiley.

> Download-Link Code Text
http://blogs.iad.zhdk.ch/codingspace/2013/09/24/fr-20-09-2013-smileys/



3 Rekursive Funktionen / Bäume

Mit rekursiven Funktionen lassen sich sehr gut Pflanzen-/Blütenähnliche Gebilde zeichnen. Doch vergiss nie die Abbruch-Bedingung ! Sonst gehts schnell ins Endlose.

> Download-Link Code Text
http://blogs.iad.zhdk.ch/codingspace/2013/09/24/fr-20-09-2013-rekursive-funktionen/